Listing of the Claims

1. (original) A method to analyze escape analysis of an application comprising:

identifying one or more methods associated with a violating condition;

parsing the one or more methods into at least one equivalence class;

identifying a first escape indicator and a second escape indicator associated with each of the at least one equivalence class; and

propagating the one or more methods based on the first and second escape indicators.

2. (original) A method as defined in claim 1, wherein identifying the one or more methods associated with the violating condition comprises identifying at least one of a new method, an additional method, and a method associated with a change in a recursive call chain.

3. (original) A method as defined in claim 1, wherein identifying the one or more methods associated with the violating condition comprises identifying one or more methods associated with at least one of dynamic class loading, native method, and reflection.

4. (original) A method as defined in claim 1, wherein identifying the one or more methods associated with the violating condition comprises identifying the one or more methods during runtime of the application.

5. (original) A method as defined in claim 1, wherein identifying the first escape indicator and the second escape indicator associated with each of the at least one class comprises identifying a bottom-up escape status flag and a top-down escape status flag.

6. (original) A method as defined in claim 1, further comprising updating the escape analysis of the application.

7. (original) A machine readable medium storing instructions, which when executed, cause a machine to:

identify one or more methods associated with a violating condition;

parse the one or more methods into at least one equivalence class;

identify a first escape indicator and a second escape indicator associated with each of the at least one class; and

propagate the one or more methods based on the first and second escape indicators.

8. (original) A machine readable medium as defined in claim 7, wherein the instructions cause the machine to identify the one or more methods associated with the violating condition by identifying at least one of a new method, an additional method, and a method associated with a change in a recursive call chain.

9. (original) A machine readable medium as defined in claim 7, wherein the instructions cause the machine to identify the one or more methods associated with the violating condition by identifying one or more methods associated with at least one of dynamic class loading, native method, and reflection.

10. (original) A machine readable medium as defined in claim 7, wherein the instructions cause the machine to identify the one or more methods associated with the violating condition by identifying the one or more methods during runtime of the application.

11. (original) A machine readable medium as defined in claim 7, wherein the instructions cause the machine identify the first escape indicator and the second escape indicator associated with each of the at least one class by identifying a bottom-up escape status flag and a top-down escape status flag.

12. (original) A machine readable medium as defined in claim 7, further comprising instructions, which when executed, cause the machine to update the escape analysis of the application.

13. (original) A machine readable medium as defined in claim 7, wherein the machine readable medium comprises one of a programmable gate array, an application specific integrated circuit, an erasable programmable read only memory, a read only memory, random access memory, a magnetic media, and an optical media.

14. (original) An apparatus to analyze escape analysis of an application comprising:

a method identifier configured to identify one or more methods associated with a violating condition;

a method parser coupled to the method identifier and configured to parse the one or more methods into at least one class;

a status identifier coupled to the method parser and configured to identify a first status indicator and a second status indicator associated with the least one class; and

a compiler coupled to the status identifier and configured to propagate the one or more methods based on the first and second status indicators.

15. (original) An apparatus as defined in claim 14, wherein one or more methods associated with a violating condition comprises at least one of a new method, an additional method, and a method associated with a change in a recursive call chain.

16. (original) An apparatus as defined in claim 14, wherein the violating condition comprises at least one of dynamic class loading, native method, and reflection.

17. (original) An apparatus as defined in claim 14, wherein the first escape indicator comprises a bottom-up escape status flag.

18. (original) An apparatus as defined in claim 14, wherein the second escape indicator comprises a top-down escape status flag.

19. (original) An apparatus as defined in claim 14, wherein the compiler is configured to update the escape analysis of the application.

20. (currently amended) A processor system to analyze escape analysis of an application comprising:

a dynamic random access memory (DRAM) configured to store one or more methods of the application; and

a processor operatively coupled to the DRAM, the processor ~~being programmed~~ to:

identify the one or more methods associated with the violating condition[[,]];

~~to~~ parse the one or more methods into at least one equivalence class[[,]];

~~to~~ identify a first escape indicator and a second escape indicator associated with each of the at least one class[[,]]; and

~~to~~ propagate the one or more methods based on the first and second escape indicators.

21. (original) A processor system as defined in claim 20, wherein the one or more methods associated with a violating condition comprises at least one of a new method, an additional method, and a method associated with a change in a recursive call chain.

22. (original) A processor system as defined in claim 20, wherein the violating condition comprises at least one of dynamic class loading, native method, and reflection.

23. (original) A processor system as defined in claim 20, wherein the first escape indicator comprises a bottom-up escape status flag.

24. (original) A processor system as defined in claim 20, wherein the second escape indicator comprises a top-down escape status flag.

25. (currently amended) A processor system as defined in claim 20, wherein the processor is ~~configured~~ to update the escape analysis of the application.